A fast method to develop an optimal operational sublevel stope design

Nelson Morales[1]
Département des génies civil, géologique et des mines
Polytechnique Montréal

Diego Mancilla, Roberto Miranda, Javier Vallejos
Advanced Mining Technology Center & Mining Engineering Department
University of Chile

KEYWORDS: mine layout optimization, Integer programming, underground mining.

[1] Corresponding author: nelson.morales@polymtl.ca
ACKNOWLEDGMENTS: The authors thank ANID's basal grant AFB180004, "Advanced Mining Technology Center".

Highlights

- A novel method to compute an optimal underground layout in which all stopes have positive economic value and are geotechnically stable.
- A new mathematical program that selects the best set of stopes from the family of valid stopes. We propose two variations of the modelling, with different constraints for controlling the geometry of the sublevels in the layout design.
- A theoretical result showing that one of the variations of the problem can be reduced to a shortest-path problem; therefore, it can be solved efficiently up to optimality.

A fast method to develop an optimal operational sublevel stope design

Abstract

Designing a sublevel stoping mine is a challenging task that requires determining the best layout in terms of economic value while respecting geomechanical constraints that limit the stopes' shape. Because of its relevance and challenging aspects, many authors have proposed methods to approximate or compute optimal stope designs or provided raw designs that can be used as a guideline. This paper follows previous approaches by approximating the shape of the stopes using the block model as support and then working in two stages: an enumeration algorithm generates all valid stopes, and then an optimization model is used to select the set of stopes with maximum value However, our approach adds several improvements. Firstly, we ensure that the stopes generated in the first stage are profitable and geotechnically stable. Secondly, the mathematical model of the second stage incorporates the organization of stopes into drifts and levels, which yields a more operational layout. Thirdly, the approach is suitable for algorithmic and theoretical contributions: We propose a fast heuristic algorithm for the general case of the model but also show that in some cases, the optimization problem reduces to finding the shortest path in an ad-hoc graph; thus, its optimum can be computed efficiently. Fourthly, we tested our approach on 4 block models involving 84,000 to over 3 million blocks, or about 53,000 to 13.5 million valid stopes, respectively. The resulting optimal layouts are not only operationally feasible, but optimal solutions can be found in less than one hour using freely available linear programming solvers or in less than a second using a shortest-path algorithm when it applies. Finally, we performed sensibility analyses to assess the variability of the value and geometric location of optimal stopes showing that the solutions are robust and that even with perturbations of 20% in the economic values of the stopes, the value and tonnage of the optimal solutions do not change more than 0.4% and that in most cases, 95% of the extracted blocks remain the same.

1 Introduction

Long-term planning of an underground mine involves three main aspects: mine design, production scheduling, and equipment selection (Musingwini, 2016). Mine design is the most critical task because it is the foundation of the other two: it determines the geometrical portions of the deposit to be exploited, i.e., the parts of the mine that need to be scheduled for extraction, and in turn, for which the equipment must be selected. However, finding a good mine design is also a complex problem that involves determining the geometry of the mine layout, which is a design problem in 3-D space. Furthermore, mine design must satisfy complex geomechanical constraints that ensure the success of the mining method of choice and the safety of the operation while assuring the richest parts of the deposit are being extracted to reach a high economic value.

In this paper, we focus on the sublevel open stoping (SLOS) method, where the stope (empty space left after mineral extraction) is extracted from drifts located at its bottom and organized in levels located at different heights. SLOS is a mining method applied to a relatively steep dip, where the ore body and surrounds are constituted by competent rock. In this method, several drives cross the orebody, and ore is produced from these drives by drilling and blasting long holes that can be inclined in any direction. The ring or pattern forms a plane perpendicular to the drive, and the holes are blasted as a unit. Figure 1 (left) presents a schematic of one level in a sublevel open stoping (SLOS) mine.

SLOS is regarded as a highly efficient and versatile mining method, enhanced by recent developments in machinery, blasting and support methods and technologies.

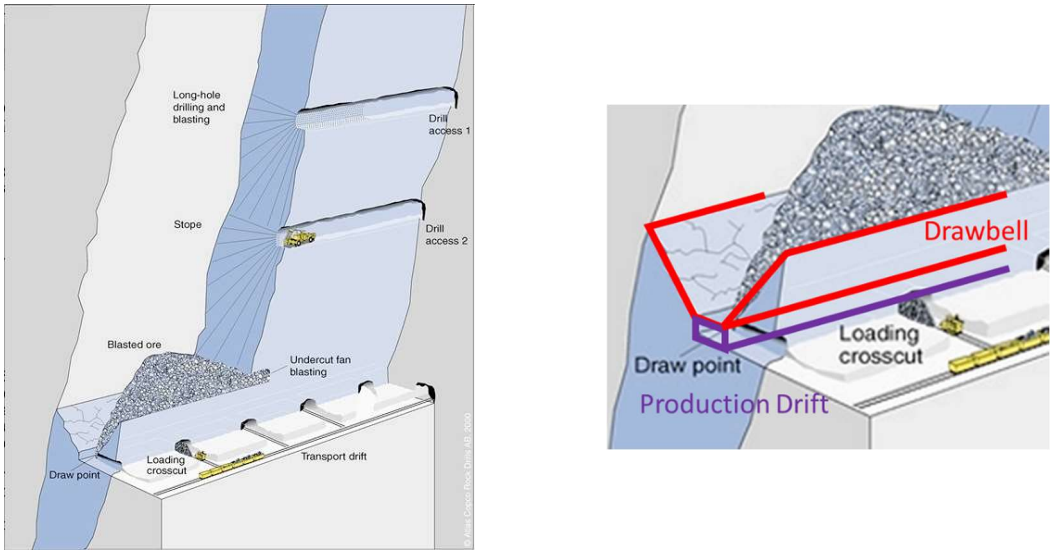


Figure 1 – (Left) Sublevel Stoping (Right) Zoom on the base of the stope illustrating the drawbell and production drift. (Atlas Copco, 2007)

Mine planners utilize the block model, a 3-D representation of the deposit as a regular array of blocks to encode the spatial distribution of relevant attributes like grades and tonnages and make decisions about mining volumes and production planning. Thus, many authors model the shape of stopes by approximating them as sets of blocks. For example, in the seminal work, Alford (1995) introduces the floating stope method, which abstracts a stope as a rectangular set of $d_x \times d_y \times d_z$ blocks. Then, he computes the economic value and tonnage of the stope as the summation of these attributes over the blocks contained in the stope. The method generates an *inner* and an *outer* envelope, and the optimal solution, which the user must determine, lies between these envelopes.

The simplicity of the floating stope method makes it easy to implement; therefore, it has become the core idea of algorithms available in commercial software. For example, Cawrse (2001) proposes to use it iteratively to optimize the design of the whole mine. Unfortunately, the floating stope algorithm and derivates are highly dependent on user expertise; thus, they are heuristic in nature (Nikbin et al., 2020).

The geometrical limitations of the floating stope method have motivated some authors to develop methods to find the overall optimal shape of the stope. A variety of studies (Bai, 2013; Nelis et al., 2016; Ovanic & Young, 1999) have addressed the problem of optimizing the shape of a stope but consider more general geometrical options. These optimization models select the blocks that will constitute the stope in such a way that they maximize the economic value but respect geometrical constraints, expressed as mathematical formulae that link the selection of individual blocks with the desired geometric properties.

These previous works focus on generating a single profitable stope that complies with a given set of geometrical constraints. However, they do not tackle the problem of computing the best set of stopes, which is known as *stope boundary optimization*. This problem consists of determining an ensemble of valid stopes that contain the maximum economic value and comply with constraints related to their relative position and the distance between them. Such a problem is regarded as analogous to the ultimate pit problem (O'Sullivan & Newman, 2015).

Another relevant problem is *layout design optimization.* This problem extends the boundary optimization to consider also their organization into levels from which their extraction will happen. Considering that the layout design is by nature an optimization problem, many authors have relied on binary or mixed-integer programming (MIP) models to address it. However, there are different approaches to model the stopes. Some

models have decision variables at the block level, i.e., their basis is a subset of the block model, and they determine the stopes by selecting the blocks to be extracted. Other authors consider aggregating blocks into predetermined sets. For example, some models work with panels that are perpendicular to the direction of an imaginary drift. In this case, the minimum volumetric unit in the optimization models is given by a panel, and the models determine the stopes as sets of these panels. Finally, other models aggregate the blocks further and work at the stope level, optimizing directly over a set of valid stopes. Some relevant papers concerning the layout design optimization problem that work at different levels of aggregation are discussed in more detail below.

Ovanic & Young (1999) present a mixed integer program to address the problem in one-dimension. Their model computes an optimal set of stopes, which are formed by consecutive panels. The method can be used to generate a stope layout by being applied to multiple panels; however, the panels and their locations are an input to the model, i.e., they must be defined in advance. Because of this, the method does not ensure global optimality.

Jalali & Ataee-pour (2004) propose an algorithm that performs several transformations of the economic block model (a block model where the economic value of each has been pre-estimated) to reduce it to a 2-D representation and then apply dynamic programming on the resulting model to compute the optimal stope layout. Unfortunately, the algorithm execution time is significant, and because the reduction to 2-D loses some information, the algorithm is better adapted to vein deposits.

Ataee-Pour (2005) and Topal & Sens (2010) propose algorithms that assume that stopes are rectangular and of known dimensions (i.e., formed as arrays of $d_x \times d_y \times d_z$ blocks). Thus, in a first stage, they compute all stopes of given dimensions and positive economic value. The second stage is a greedy algorithm that, in each iteration, adds the stope with highest value that is compatible with previously selected stopes (i.e., such that it does not overlap). These greedy algorithms exhibit short computation times but cannot ensure optimal solutions. This issue is partially addressed by Sandanayake et al. (2015), with a more sophisticated approach in which the algorithm considers several possible initial stopes which induce different sequences for adding stopes greedily. Still, the algorithm is a heuristic that cannot ensure optimality (Erdogan & Yavuz, 2017).

Grieco & Dimitrakopoulos (2007) extend Ovanic & Young (1999) and present an integer program that computes all the stopes for all the panels simultaneously. Their emphasis is on uncertainty; therefore, they incorporate additional constraints for controlling the level of risk to be accepted. They apply their method to a real case to generate different layouts depending on the risk parameter.

Nikbin et al. (2019) present an optimization model that works at the block level. The resulting model is complex and hard to solve; therefore, the authors develop heuristics which in their case study are within 0.6% of the optimal solution. The model determines a set of rectangular stopes but offers little control of the resulting geometry. Also, because it constructs the stopes from the blocks, it is difficult to extend their approach to incorporate non-linear properties of the resulting stopes, which we discuss later in this section.

Nikbin et al. (2020) describe a heuristic method based on dynamic programming and a greedy algorithm that work at different z coordinates. Dynamic programming is used to determine optimal solutions to a one-dimensional problem, which are then combined using a greedy procedure.

It is interesting to note that for less aggregated modeling (block and panel levels), models tend to have more complex formulations because they must express the shape of the stopes using mathematical constraints. Conversely, more aggregated modeling (stope level) allows simpler mathematical formulations because the complexity of the feasible stopes' shapes is "hidden" in the input. However, this implies that "stope-based" models require a previous stage for computing the set of valid stopes. Moreover, there is the risk that the size of the input increases considerably and may become intractable because the number of stopes can be exponential with regard to the number of blocks. In fact, several works generate the set of *valid stopes* as

rectangular sets of blocks with a positive economic value. However, because of the large size of the family of valid stopes, these works apply some sort of greedy algorithm to compute the solution iteratively or to construct several potential solutions that are then compared. Indeed, Hou et al. (2019) and Little et al. (2013) acknowledge difficulties in computing good solutions efficiently.

Despite the numerical issues described above, stope-based models have a significant advantage. Indeed, some properties of the stopes, like stability or dilution, can strongly affect the value or even validity of a stope (Grieco and Dimitrakopoulos ,2007); however, these attributes are not additive like grades or tonnages, i.e., they cannot be computed using linear functions of attributes of the blocks or panels within the stope. Because of this, block and panel-based models may have to deal with non-linearity, work only with approximated values, or ignore these issues completely. A stope-based model does not suffer from these inconveniences because all the relevant attributes can be evaluated before the optimization process begins, i.e., the optimization model can use more accurate information.

Finally, it is worth noting that some authors have gone beyond modeling only the volume to be extracted and have incorporated aspects related to production scheduling in the layout design. Little et al. (2013) used a stope-level approach combined with an integer linear optimization model that not only determines the set of stopes for extraction, but also establishes when each stope should be extracted to maximize the net present value (NPV). Their model considers production and backfilling capacities as well as geometrical constraints on the stopes. Hou et al. (2019) extended this approach to also consider the development of access ramps.

In this paper, we rely on a stope-based formulation. Our methodology, as others, first computes the set of all valid stopes and then solves an optimization problem to find the best stope layout. However, we incorporate the following relevant contributions with regard to the previous models proposed:

- In our approach, the stopes in the optimal underground layout have positive economic value (as in previous works), but their shape is not constrained to be rectangular. Moreover, before admitting a stope as valid, we utilize a stability graph (Mawdesley et al., 2001) to filter out stopes that are geotechnically unstable. This is an example of using the modeling advantage of stope-based modeling that has not been exploited before.
- The optimization problem is new and more flexible, allowing several degrees of control over the geometry of the resulting solution. In this work, we propose a *free* variation in which production drifts can be located "anywhere" (as long as stopes remain compatible) and a *constrained* variation where the drifts must be organized into sublevels. (See Figure 3 in Section 1.1 for more details.)
- The mathematical model leads to efficient algorithms:
    - For the "constrained" variation of the problem, we prove that it can be solved optimally in $O(N^2)$, where N is the number of stopes. In the larger block model studied in this paper, which contains more than 13 million stopes, the proposed algorithm took 0.43 seconds to find the optimal solution.
    - For the "free" variation of the problem, we propose an algorithm that can find feasible solutions quickly. For example, in the case mentioned above, the heuristic found a solution which was within approximately 20% of the optimum in 17 seconds.
    - When using commonly available linear optimization solvers, the computation times required to find the optimum of our model scale well. The computation time to find the optimum in the 13 million stope case is approximately 45 minutes on a desktop computer.

Finally, while in this paper we present the problem in the context of sublevel open stoping (SLOS), some of the ideas could be applied directly or adapted to other underground methods. For example, our method could be applied directly to a sublevel caving mine. Similarly, in this work we use the stope-based model to ensure that the stopes are stable, but this could also be used to have a better estimation of costs by linking the size of the stopes with equipment productivity or economic value by introducing other aspects like

dilution and over excavation, which are non-linear. However, the additional attributes of the stopes need to be computed efficiently to keep the approach practical.

1.1 Description of the model

In the case of SLOS, as well as other mining methods, stopes cannot be arbitrarily located in 3-D space. Instead, they follow the development of levels and production drifts that are first used for access and extraction of the stopes. For this reason, we impose certain geometrical constraints on the selection of drifts, such that the resulting geometries are more realistic and amenable for later design of the mine. In this regard, we consider two aspects: first, that the stopes must be organized into drifts, and second, that drift locations cannot be arbitrary.

Organization of stopes into drifts. To model this, we will assume that the drifts follow the y direction of the block model (arbitrary directions can be addressed by rotating the model); thus, we encode potential location of drifts as (x, z) pairs. The model determines which drifts (i.e., (x, z) coordinates) need to be developed and the set of stopes belonging to that drift that must be extracted. More specifically, we consider that a stope potentially belongs to a drift if a drift segment can be constructed to reach it directly or it can be reached by a valid sequence of stopes in increasing order of y. This is depicted in Figure 2, which provides a plan view of a sequence of stopes constituting a drift. The squares represent blocks in the block model, with colored blocks (orange, green, light blue and yellow) corresponding to four stopes that have been selected for construction. In general, a stope can be reached through many paths, i.e., the yellow stope at the eastern location could be reached by selecting different orange, green and blue stopes.

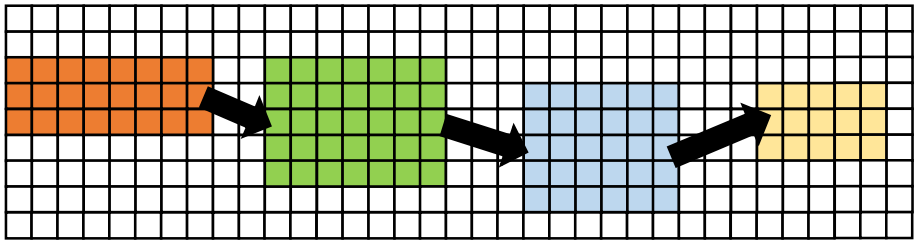


Figure 2 - Plan view of a drift with 4 stopes, each with different size.

Drifts organized into levels. We consider two variations of the model for controlling the geometry of the solutions. The first variation (*free drift location*) permits drifts to be located anywhere if there is some minimum distance between them. A second variation (*constrained drift location*) requires the drifts to be organized into levels with a minimum vertical distance between levels. Figure 3 depicts these two approaches in a conceptual case of a block model, which is 14 blocks wide and 15 blocks high. On the left, the figure depicts a feasible solution for the free drift location case, consisting of five drifts. The black blocks represent the drifts themselves and the colored blocks the blocks to be extracted (of the first stope of the drift). On the right, a feasible solution to the constrained drift location case is shown. In this case, drifts are grouped into levels $z = 3$ and 9. It is worth noting that the figure is conceptual and aims to illustrate how different the models can be. Indeed, while the solution on the right is feasible for the free drift model on the left, the converse is not true.

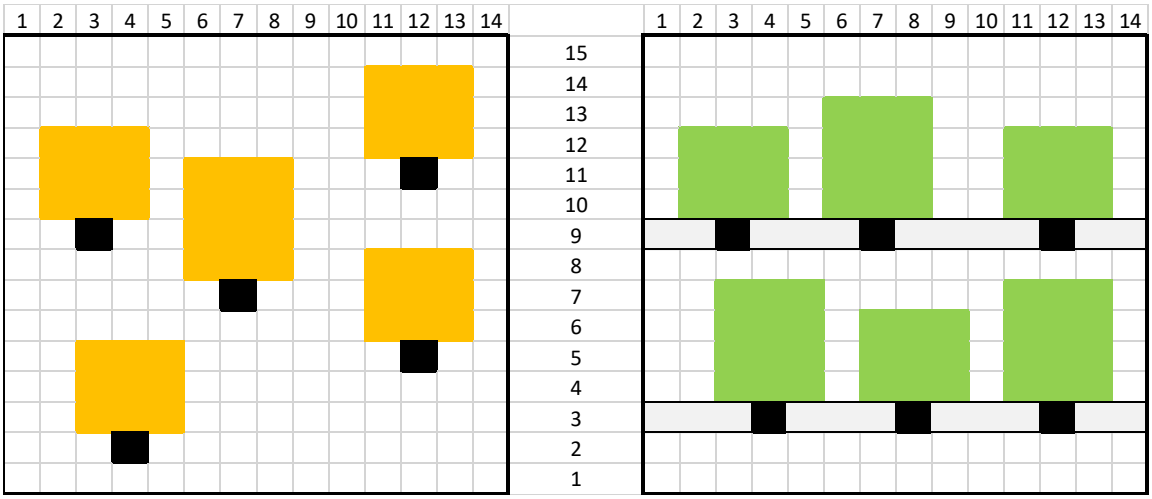


Figure 3 - An XZ-view comparing two feasible (not necessarily optimal) solutions. Black blocks represent drifts and colored blocks correspond to stopes. On the left, a free drift variation with 5 drifts. On the right, a solution of the constrained drift variation with two levels and six drifts.

We acknowledge that the constrained drift location is perhaps too restrictive; however, we emphasize that the resulting levels are an output of the model and not imposed by the user. In fact, it would be easy to extend the formulation, for example, to limit the number of levels or to limit the difference in heights between z coordinates of the drifts to be constructed.

2 Mathematical formulation and theoretical results

In this section, we introduce the mathematical notation and optimization model proposed to find the best layout of the mine, i.e., for selecting the drifts and stopes to be constructed. We also present theoretical results that justify the algorithms proposed for solving the optimization problem. We first address a one-dimensional version of the problem, corresponding to a single drift. The general case is constructed from it.

As previously noted, our approach is stope-based, i.e., the model does not work directly with blocks but assumes that the stopes have been computed in advance along with their relevant attributes. Moreover, we assume that it is possible to determine whether a potential stope is stable and, for any pair of stopes, if they are incompatible (for example, if the stopes overlap, they are incompatible).

2.1 Stope layout optimization (one drift case)

In the one drift case, we have a set Λ of stopes that belong to a potential drift. For stope $\ell \in \Lambda$, let $v(\ell)$ be its economic value. We assume that the compatibility between two stopes (i.e., if they both can be constructed) can be easily checked (for example, stopes that overlap or are too close to each other are not compatible). Thus, we denote $\Gamma = \{\{\ell_1, \ell_2\} \in \Lambda : \ \ell_2 \text{ is not compatible with } \ell_1\}$.

With these definitions, we see that the problem of finding the set of stopes in a drift that have maximum value can be formulated as a binary linear program as follows. Let u_ℓ be a binary variable such that $u_\ell = 1$ if and only if the corresponding stope is extracted. We name the resulting optimization problem as the "one drift slope design problem," or $(1 - SDP)$ for short. This problem can be formulated as follows.

$$
\begin{aligned}
(1-SDP) \quad max \quad & \sum_{\ell \in \Lambda} v(\ell) u_\ell & \\
& u_{\ell_1} + u_{\ell_2} \leq 1 & \forall \{\ell_1, \ell_2\} \in \Gamma \\
& u_\ell \in \{0,1\} & \forall \ell \in \Lambda
\end{aligned}
$$

From the formulation, it is also clear that the problem is completely determined by the graph $G = (\Lambda, \Gamma)$ with weight function v defined on the vertices, and in fact, $(1 - SDP)$ corresponds to finding an independent set of maximum weight G, which is NP-Hard (Garey & Johnson, 1978). Fortunately, the hardness of a maximum independent set comes from the arbitrary structure of the graph. In fact, we will later show that $(1 - SDP)$ is equivalent to finding the longest path in an auxiliary graph and that it can be solved efficiently.

2.2 Stope layout optimization (multiple drifts)

To model this case, we generalize the previous formulation to determine not one, but several drifts each containing many stopes. In this formulation, the model chooses which drifts and stopes are to be constructed such that the total economic value is maximized, each individual stope is stable, and no two incompatible stopes are constructed. Given the aforementioned and because we assume that the drifts follow the y axis, we can enumerate the set of possible drifts by the (x, z) coordinates of its left-upper block. We will denote $XZ = \{(x, z)\}$ the set of valid locations for the drifts.

As drifts are indexed by coordinates, we observe that each drift (x, z) induces an instance of $(1 - SDP)$ with its own stope and incompatibility sets, which we can write as Λ^{xz} and Γ^{xz}, respectively. Notice that Γ^{xz} represents the incompatibility between stopes within the drift indexed by coordinates (x, z); therefore, we need to extend this to establish incompatibility constraints between drifts of different coordinates. We consider two variations:

- *Free drift location.* In this case we consider that two drifts are incompatible if they contain stopes that are incompatible with each other. Then, a drift with coordinates $m = (x, z)$ is compatible with coordinates $m' = (x', z')$ if and only if all stopes in m are compatible with all stopes in m'. (Recall that it is always possible to evaluate if two stopes are compatible.)
- *Constrained drift location*. In this case, we consider that two drifts with the same compatibility as before (two drifts are compatible if all their stopes are compatible), plus the constraint that drift $m = (x, z)$ is compatible with any drift $m' = (x', z')$ only if $|z - z'| = 0$ or $|z - z'| \geq D_z$, for some distance D_z. In other words, we observe that selected drifts must have all their stopes compatible and either belong to the same level or to levels that are far enough from each other.

In any of the cases, we denote as $\hat{\Gamma} = \{\{(x, z), (x, z')\}\}$ the incompatibility set for drifts and introduce the following optimization binary variables $w_{xz} = 1$ if and only if a drift (x, z) is constructed.

As mentioned before, for each value of $(x, z) \in XZ$ there is a corresponding instance of $(1 - SDP)$, thus, if we denote the corresponding variables in the formulation as u_ℓ^{xz}, this allows us to introduce the "Multi-drift stope design problem" or $(n - SDP)$ for short. The model can be written as

$$(n - SDP) \quad max \quad \sum_{xy \in XZ} \sum_{\ell \in \Lambda^{xz}} v(S_\ell) u_\ell^{xz} \tag{1}$$

$$u_{\ell_1}^{xz} + u_{\ell_2}^{xz} \leq 1 \qquad \forall (x, z) \in C, \forall \{\ell_1, \ell_2\} \in \Gamma^{xz} \tag{2}$$

$$w_{xz} + w_{x'z'} \leq 1 \qquad \forall \{(x, z), (x', z')\} \in \hat{\Gamma} \tag{3}$$

$$u_\ell^{xz} \leq w_{xz} \qquad \forall (x, z) \in XZ, \forall \ell \in \Lambda^{xz} \tag{4}$$

$$u_\ell^{xz}, w_{xz} \in \{0,1\} \qquad \forall (x, z) \in XZ, \forall \ell \in \Lambda^{xz}$$

Equation (1) is the objective function and corresponds to the total economic value. Equation (2) prevents two incompatible stopes within the same drift from being selected. Equation (3) establishes the incompatibility constraint between drifts. Finally, Equation (4) indicates that only stopes belonging to selected drifts can be constructed.

We observe $(n-SDP)$ can be very large and that it is a specific case of the weighted maximum independent set; thus, it may be difficult to solve up to optimality. Because of this, we discuss some algorithmic approaches in the next section.

2.3 Solution algorithms

Because $(n-SDP)$ is a binary linear problem, it is theoretically possible to solve it using the Branch and Bound (BnB) algorithm (Land & Doig, 1960). This algorithm works by solving the integer relaxation of the problem to generate feasible solutions and upper bounds that can establish the quality of the feasible solutions found so far. BnB is the default method for solving mixed integer or binary integer problems because it is available in most academic and commercial optimization software. Another important feature of BnB is that it permits an optimality gap $\varepsilon \geq 0$ to be set, such that the algorithm ends either with a proof of unfeasibility or a feasible solution that cannot be improved beyond $100 \cdot \varepsilon$%. Therefore, setting $\varepsilon = 0$ ensures that the algorithm reports an optimal solution if it exists.

Unfortunately, as mixed integer linear problems can be NP-Hard in general, BnB is potentially inefficient or impractical in general. Hence, it is convenient to consider some algorithms that are specifically designed for exploiting the structure of the problem. In section 2.3.1 we address the constrained drift location case and show that it can be solved efficiently by reducing it to the shortest-path problem. In section 2.3.2, we address the free drift location for which we propose a simple heuristic algorithm.

2.3.1 Solving the constrained drift location case by means of a shortest-path algorithm

In the case of $(n-SDP)$ with a constrained drift location ($(1-SDP)$ is a particular case of it), we show that the problem can be reduced to a longest-path problem in an auxiliary graph that contains no cycles. Therefore, it can be solved efficiently because this problem in turn can be reduced to a shortest-path problem (Sedgewick & Wayne, 2011).

Let us first address the one-dimensional case and assume that y coordinates are ordered from *left* to *right*. Therefore, any pair of compatible stopes are such that one is located at the left of the other. This is illustrated in Figure 4, where an arbitrary stope ℓ is drawn at the center, and compatible stopes are represented in segmented lines (these stopes are compatible with stope ℓ, not with each other).

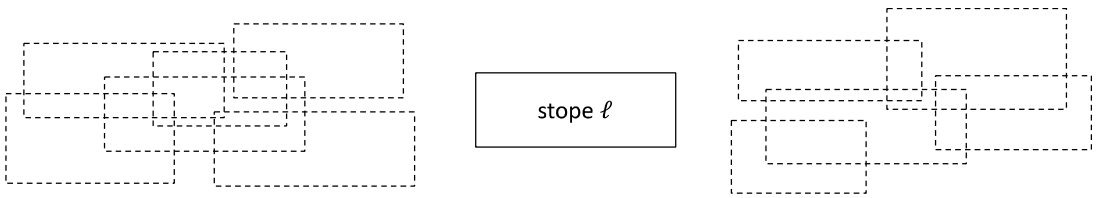


Figure 4 - A conceptual example of a stope (solid line, at the center) and all stopes that are compatible with it (in segmented lines).

We construct the following auxiliary graph $G = (V, A)$

- The set of vertices $V = \{s\} \cup \Lambda$, where s is an artificial vertex named the *root*.
- The set of arcs is $A = \{(s, \ell) : \ell \in \Lambda\} \cup \{\ell_1, \ell_2 \in \Lambda : y(\ell_1) < y(\ell_2) \wedge \{\ell_1, \ell_2\} \notin \Gamma\}$, i.e., the root is connected to all vertices (stopes), and two stopes are connected if one is at the left of the other and they are compatible.
- If arc $(u, v) \in A$ connects to vertices, then the length of the arc is $\delta(u, v) = v(S_\ell)$, the value of the destination stope. (Recall that by construction all economic values are positive.)

It follows that an independent set of stopes (i.e., a set of stopes that are all compatible with each other) corresponds to a path starting in the root s. Moreover, the economic value of the set of stopes is exactly the

length of the path. Thus, finding the longest path that starts in s is equivalent to finding the set of independent stopes of maximum economic value.

To conclude the proof, we observe that because Graph G is acyclic, the longest path can in turn be calculated as a shortest path by replacing the distances with $\delta'(u,v) = \Delta - \delta(u,v)$ where Δ is any sufficiently large value such that all δ' values are positive (for example $\Delta = 1 + \max_{(u,v)} \delta(u,v)$). In this way, the new distances are all positive and the Dijkstra algorithm can be used to compute the shortest path (Dijkstra, 1959) in $O(|\Lambda|^2)$ operations.

For the case of $(n-SDP)$, we observe that the previous reduction to a longest path problem works because to verify that the set of selected stopes is valid, it suffices to check the compatibility in a sequential way, where stopes are organized as a unique path from left to right. This idea in fact can be generalized to $(n-SDP)$ with constrained drifts. In this case, drifts can be ordered in ascending order of x coordinates, and then, since all drifts in a level are compatible with all drifts in the next one, they can also be ordered by z. Therefore, $(n-SDP)$ can be solved efficiently in the case of the constrained drift formulation. A detailed proof of this can be found in the appendix.

2.3.2 Heuristic for the free drift case

The previous reduction does not apply in the free drift location case because if drifts can be positioned at arbitrary coordinates, it is not possible to sort them such that the independence of the selected stopes can be verified by checking compatibility of subsequent stopes in a path. As a result, computing a longest path produces an independent set of stopes, i.e., a feasible design, but not necessarily optimal.

Considering the above, we propose using an adapted version of the Bellman-Ford's algorithm (Bellman, 1958; Ford, 1956). The complexity of this algorithm when working on a directed graph $G = (V, A)$ is $O(|V||A|)$. Considering that the complexity of Dijkstra's is $O(|V|^2)$, and there are more arcs than vertices, the algorithm that we propose is slower for large graphs. However, Bellman-Ford's algorithm can work with negative distances in graphs that contain cycles, which is necessary in our case.

Our adapted version of the algorithm is straightforward. We illustrate it in Figure 5. In fact, except for the line in red, it corresponds exactly to the original algorithm. This red line forces that when updating the distances at the evaluation of arc (u,v), only compatible stopes (represented by v) are considered.

```
1   Let Q = {s}
2   For each vertex v ≠ s ∈ V:
3       d(v) := ∞
4   d(s) := 0
5   While Q is not empty, do:
6     u := Pull(Q)
7     For each edge {u,v} ∈ E do:
8       If d(u) + w(u,v) < d(v) then
9         If not IsCompatible(Predecessors(u), v) then go to 7.
10        d(v) := d(u) + w(u,v)
11        If v ∉ Q then Push(v, Q)
```

Figure 5 - Adapted Bellman-Ford algorithm

3 Numerical Experiences

In this section, we present numerical results obtained when applying our model and solution techniques in two copper mines located in Northern Chile. We first describe the instances in terms of economic parameters, geotechnical modeling, and block models, and then present the results and their analyses.

3.1 Description of the instances

The parameters used to evaluate the economic values of individual blocks are summarized in Table 1 (copper price, metallurgic recovery, mining cost, etc.). These values, plus the construction costs for the drifts are used to compute the economic value of the stopes.

It is worth noting that while Mine A is an actual mine, the only attributes that were used were the block locations and copper grades; thus, the density was assumed to be constant and the geomechanical model was constructed ad-hoc for the study. This is not the case for Mine B, for which all data was available to perform the studies.

Table 1: Parameters for economic evaluation of blocks

Parameter (Symbol)	Unit	Mine A	Mine B
Price (P)	USD/lb	3.5	
Selling Cost (C_s)	USD/lb	0.25	0.5
Mine Cost (C_m)	USD/ton	20	25
Processing cost (C_p)	USD/ton	12	12
Metallurgical Recovery (R)	%	80	85
Conversion Factor (F)	lb/ton	2,204.62	
Development cost (D)	USD/m	4,000	3,700

If g_i is the copper grade of the i-th block and t_i is its tonnage, then its economic value is

$$V_i = (P - C_s) \cdot g_i \cdot t_i \cdot R \cdot F - (C_m + C_p) \cdot t_i$$

It is important to observe that the mining cost C_m does not include the development required for the drift segment located under the stope. Indeed, the economic value of a stope will be computed as the addition of the economic blocks contained in it minus the development cost of that drift segment.

To test the approach and stress the model, we utilized the block models of the mines to generate additional instances by splitting the blocks into 8 smaller ones. For these blocks, we considered the same stability values and assigned value and tonnage equal to one eighth of the original blocks.

The resulting instances are reported in Table 2, which contains the following for each instance: an identifier (ID), the size of individual blocks, the total number of blocks in each direction (NX, NY, NZ), the total number of blocks in the block model, the desired stope dimensions, and the minimum distance required between stopes in the vertical dimension (crown pillar) and between stopes in the same drift (min. inter-drift distance), and the minimum size of pillars for stopes in the same drift (min. pillar length).

Table 2 Summary of block model instances

ID	Block Sizes (m)	NX	NY	NZ	# Blocks	Stope width (m)	Stope height (m)	Stope length range (m)	Drawbell height (m)	Crown pillar height (m)	Min. inter-drift distance (m)	Min. pillar length (m)
A0	10x10x10	36	73	32	84,096	30	30	30-50	20	10	20	20
A1	5x5x5	72	146	64	672,768	30	30	30-50	20	10	20	20
B0	10x10x16	139	172	17	406,436	30	48	40-80	32	16	10	10
B1	5x5x8	278	344	34	3,251,488	35	48	40-80	32	16	10	10

3.2 Implementation

All code was implemented in Python (v3.6) and the following publicly available libraries: *pandas v1.1.4* (McKinney, 2010) for storing and managing the block model data, *numpy v1.19.0* (Hoyer, et al., 2020) for efficient computation on the data, *networkx* v2.5 (Swart, Hagberg, Schult, & Pieter, 2008) for the implementation of the Bellman-Ford algorithm, its modified version, and the implementation of the maximum weighted clique algorithm, *mayavi* v4.7.2 (Ramachandran & Varoquaux, 2011) for visualization, *pathos* v0.2.7 (McKerns, Strand, Sullivan, Fang, & Aivazis, 2011) for parallelizing the generation of stopes, and *pulp* v2.3.1 (Mitchell, O'Sullivan, & Dunning, 2011) to model $(n - SDP)$ and as interface to the optimization solver. We utilized CBC (Forrest, et al., 2015) for finding optimal solutions of $(n - SDP)$.

3.3 Generation of the valid stopes

The first step of the methodology is common to both variations of the problem and corresponds to enumerating all possible stopes, computing their economic value, and discarding non-profitable and non-stable stopes.

To generate the stopes, we constructed them as sets of blocks; however, we incorporated the drawbell and the segment of drift located below the stope. This allows for a better approximation of the final shape as well as the economic value. In this way, costs can be approximated more accurately because, for example, drilling and blasting costs of drawbells are relatively higher. Specifically, we utilized the concept of *pattern*, which encodes the width and height of the stope as well as the bell and the drift segment. As in the case of panels in other publications, a pattern is repeated several times to generate stopes of different lengths. The concept and the process are illustrated in Figure 6, where (a) represents a pattern, (b) its approximation using blocks, and (c) the generation of a stope by repeating the pattern.

In the applications presented in this work, we used only one pattern per instance; however, several could be utilized in general to produce stopes of different sizes.

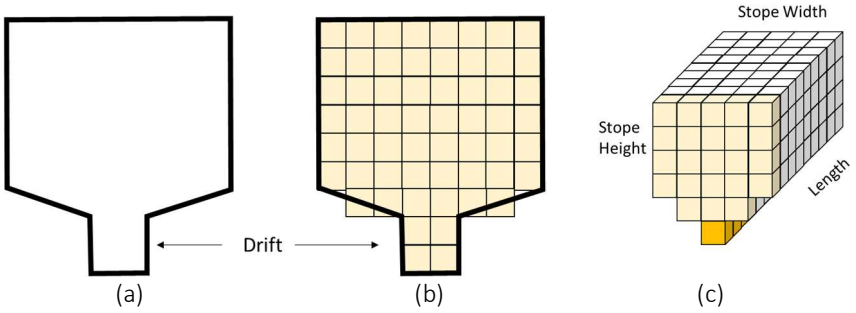


Figure 6 - (a) Actual stope and drift profiles. (b) Block pattern to approximate the stope. (c) Isometric view of a stope.

The successful application of SLOS requires that the extracted cavity be stable; therefore, many studies have focused on supporting the design process of stopes. Mathews et al. (1981) developed the first stability graph, which was based on the study of 26 case stories. Other authors, for example Potvin (1988), Sourineni (2010), then expanded the database and updated the model. The database reported in Mawdesley et al. (2001) contains more than 400 cases of stopes in North America, Australia, and England.

It follows that a stability graph determines a function $\omega(\cdot)$ which can be used to determine the stability of each stope individually. Using this, we can filter out all stopes that are not stable before any optimization process begins. This ensures that the final solution does not propose unstable stopes but also has the potential to reduce the total number of stopes in the optimization process which shortens its running time.

Table 3 reports the results obtained in the four block models in terms of the total number of potential stopes (i.e., that fit within the block model based on their dimensions) and valid stopes (i.e., that are both stable and have positive economic value). The column "# Valid Drifts" contains those drifts with at least one valid stope. Similarly, the column "# Valid Levels" corresponds to levels that contain at least one valid drift.

Table 3 Summary of results for stope generation

ID	# Potential Stopes	# Valid Stopes	# Valid Drifts	# Valid levels
A0	53,466	5,714	241	27
A1	715,260	74,635	955	52
B0	949,395	28,620	491	12
B1	13,522,464	416,408	2,029	24

3.4 Stope optimization results (constrained drift location case)

In this case, any shortest-path algorithm is guaranteed to find the optimal solution for the problem; hence, we used the Bellman-Ford implementation available in the *networkx* package without modification. Table 4 reports, for each instance, the number of stopes, drifts and levels selected in the optimal solutions, as well as the total stope value and the execution time, both for the algorithm and the exact solution.

Table 4 Summary of results for the constrained drift location case

ID	# Stopes	# Drifts	# Levels	Total Stope Value (MMUSD)	Execution Time (s)
A0	14	8	4	74.92	0.01
A1	14	8	4	73.25	0.06
B0	87	21	2	469.61	0.06
B1	85	21	2	468.97	0.43

As the results show, even for the largest instance, the algorithm can find the optimal solution in less than half a second. The results also show that the solutions are consistent in each block model, i.e., the total number of stopes, drifts and levels is the same, for A0 and A1, and for B0 and B1, respectively. The economic value of the optimal solution with the highest fidelity A1 is slightly smaller than that for A0 (the same applies when comparing B1 and B0). This is most likely due to a better approximation of the drift and bell shapes.

3.5 Stope optimization results (free drift location case)

In this case, it is not possible to reduce the problem to a longest-path instance; therefore, we used the adapted version of the Bellman-Ford algorithm presented in Figure 5 and compared it with the results from the CBC solver. The optimality gap for CBC was set to zero, i.e., the solutions found by the BnB implementation in CBC are optimal.

Table 5 reports the solutions obtained by the two algorithms. It indicates the instance ID, the number of stopes, drifts and levels, the economic value of the solutions, and the execution time. Because the binary linear approximations are optimal, the table also reports the gap and time improvement with regard to the optimal solution of the binary linear program.

Table 5 - Summary of results for the free drift location case

ID	Adapted Bellman-Ford Algorithm					Binary Linear Program					GAP (%)	Time Gained (Factor)
	# Stopes	# Drifts	# Levels	Total Stope Value (MMUSD)	Exec. Time (s)	# Stopes	# Drifts	# Levels	Total Stope Value (MMUSD)	Exec. Time (s)		
A0	13	8	8	72.12	0.27	16	9	9	82.80	32	12.9%	120
A1	15	9	6	74.68	3.63	16	9	9	79.74	1,917	6.3%	528
B0	87	21	6	517.99	1.20	99	25	9	654.04	28	20.8%	24
B1	94	23	9	560.32	17.00	108	28	10	676.54	2,369	17.2%	139

Overall, we see that the adapted algorithm is very fast, with the longest execution time being of 17 seconds, reaching a speedup from 24X to 528X, if compared to the linear optimization solver. However, the optimality gap can also be relatively large: up to 20.8% in the worst case, which suggests using the heuristic to provide an initial solution for BnB.

It is worth noting that the solutions reported in Table 5 are comparable in value with those of the constrained drift case; however, the optimal solutions show that the free drift solutions can contribute significantly to the value of the layout, with an increase between 13% and 25%, depending on the instance. However, these benefits require that the selected stopes be distributed over more levels. For example, in the A0 and A1 cases, the optimal solutions for the constrained drift location case distribute stopes in only 2 levels, while they require 9 in the free drift location variant. For B0 and B1, the increase in the number of levels is even more, from only 2 levels up to 10.

3.6 Comparison of layouts

In this section, we present some isometric views of the solutions found in the different cases.

Figure 7 presents a comparison between the results obtained in the constrained and free drift location model variations for the A0 and A1 models. Figure 8 presents the analogous information for models B0 and B1. As shown in Tables Table 4 and Table 5, the free location variant captures more value by generating more stopes; however, this is at the expense of a more irregular location of the drifts, i.e., more complexity in the design and subsequently in mine operation. Surprisingly, we observe that the designs proposed by both variations of the problem look relatively similar; however, as expected, the constrained drift location distributes the stopes into few levels, making for a simpler design. For example, in the case of A0, A1, the extra value comes from adding two more levels, which contain one stope each; thus, most likely these stopes are not profitable if all development costs are considered. In the case of B0 and B1, the model manages to retrieve additional value by changing the location of levels across the y coordinate, extending from only 2 levels (with a total of 85-87 stopes) to 9-10 levels (with 99-108 stopes). The resulting layout may be harder to design; however, the results suggest that both variations could be combined, for example, by splitting the block model in an XY-plane and allowing a different definition of levels in each section to adapt.

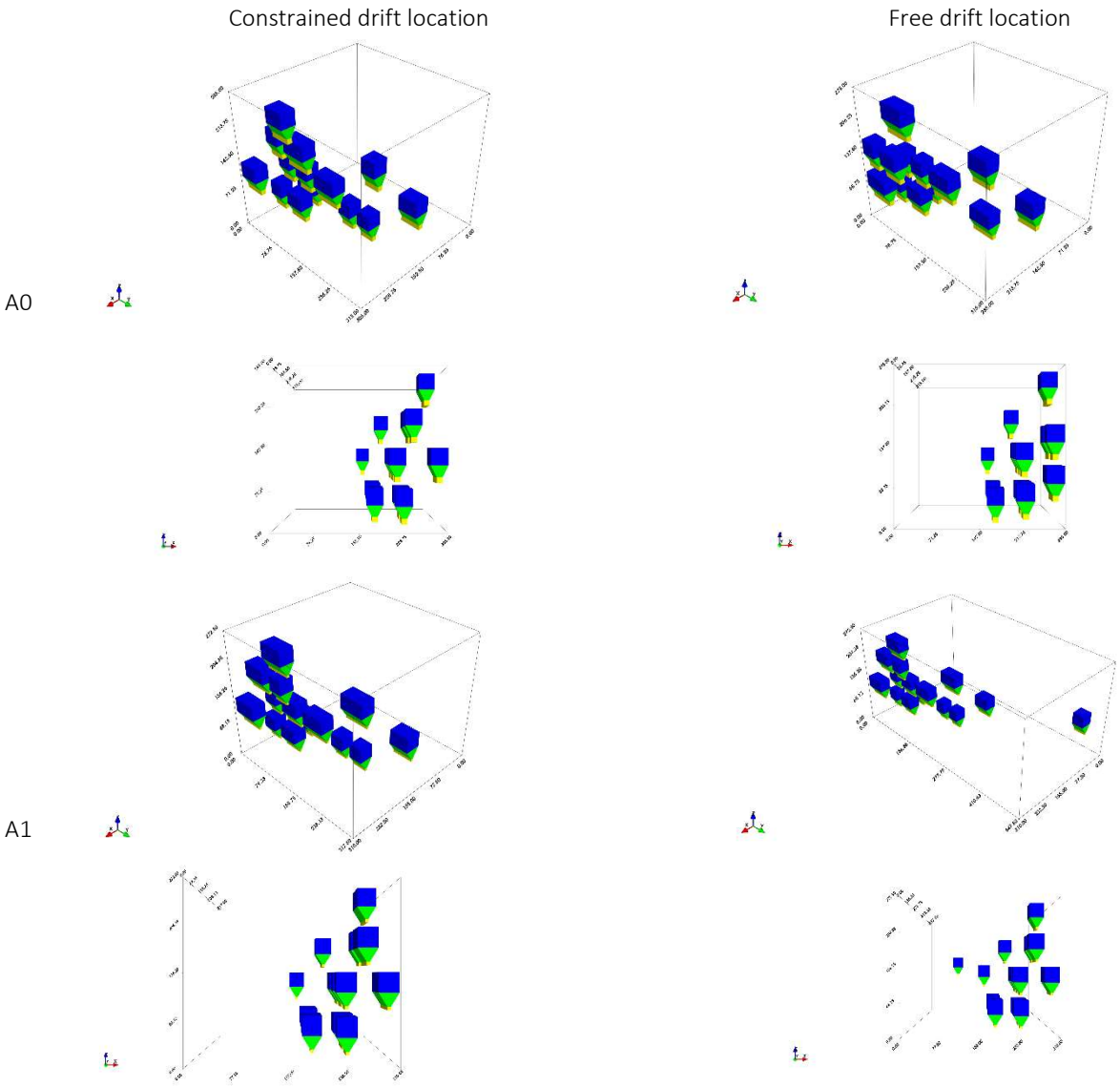


Figure 7 Isometric views of the designs generated by the algorithm for instances A0 and A1

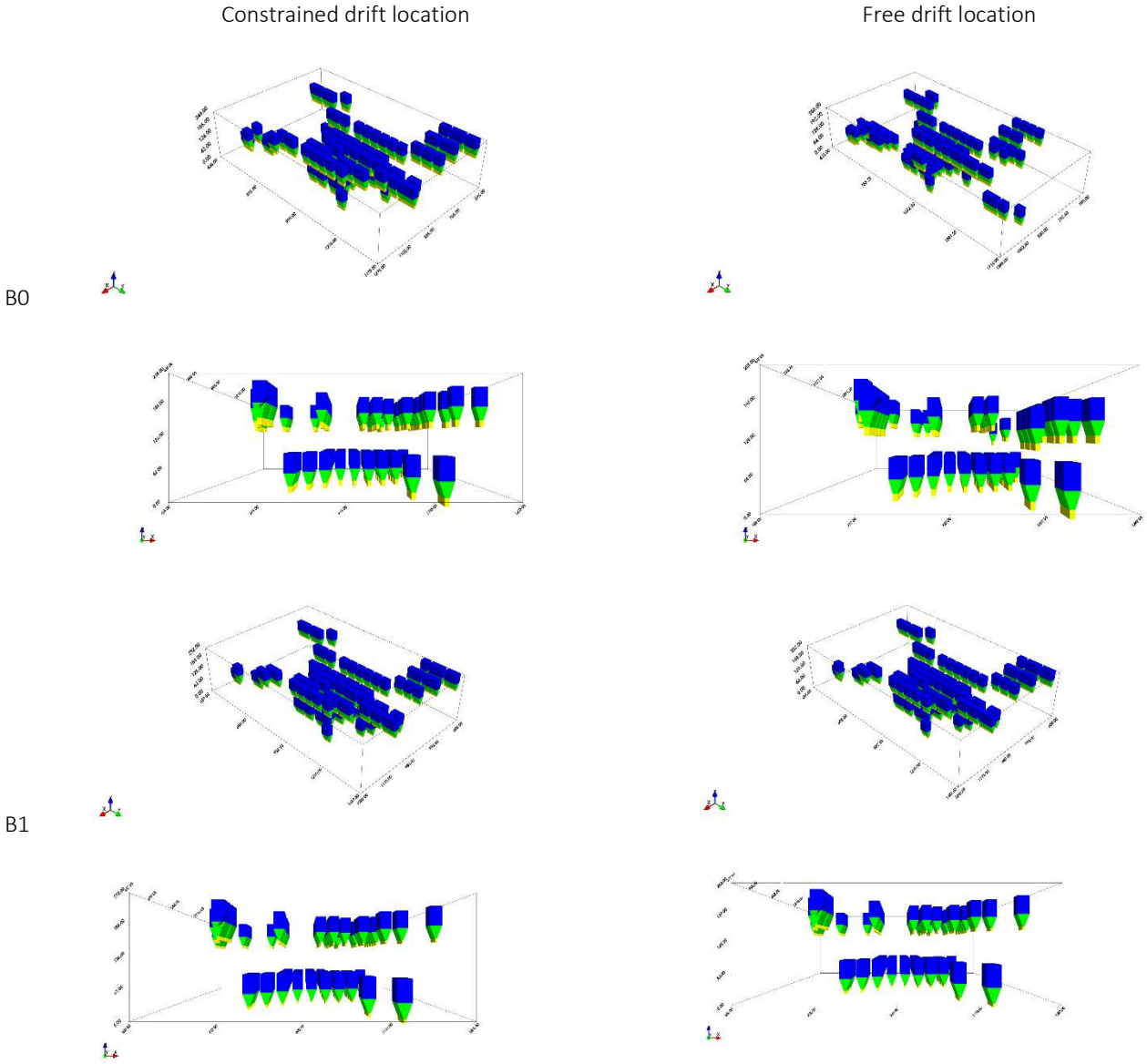


Figure 8 Isometric views of the design generated by the algorithm for instances B0, B1 and B2

Figure 9 presents side by side the designs obtained by the adapted algorithm and the optimal solutions in the free drift case. Indeed, we observe that the adapted algorithm generates solutions that are very similar. Overall, we observe that the proposed layouts are not too different; however, the optimal solution obtained by solving the monolithic model finds a layout with a higher density of stopes, increasing from 87 to 99 stopes in the B0 case and 94 to 108 in B1. This is due to the greedy nature of the adapted algorithm.

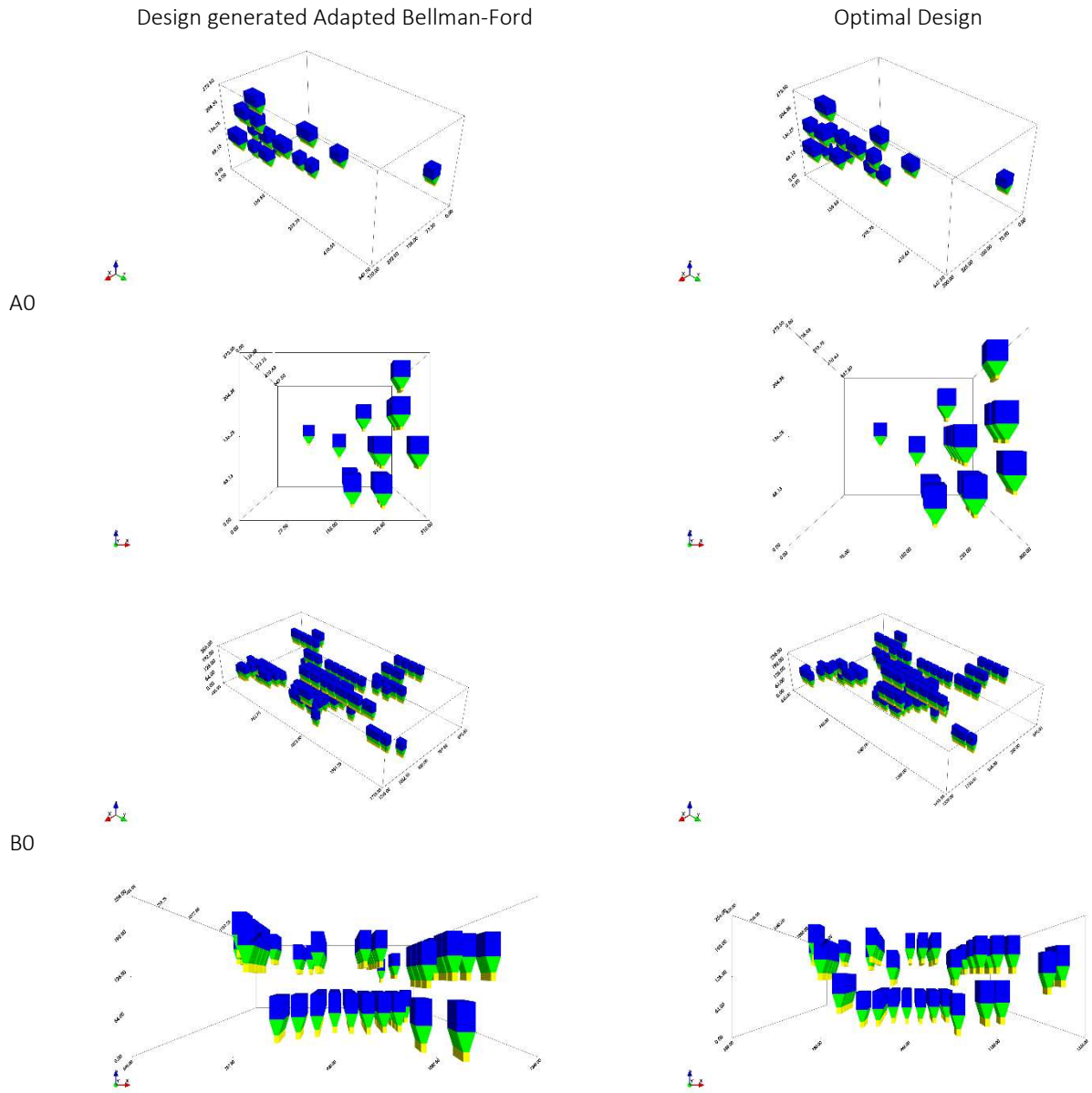


Figure 9 Comparison of designs obtained by the algorithm and exact solution (isometric views)

3.7 Sensibility Analysis

In this section we take advantage of the short computation times to analyze the robustness of the solutions obtained. For this, we consider random perturbations of the economic values of the stopes and analyze how the optimal solution changes for the 2 larger block models, i.e., A1 and B1.

To construct the scenarios for the sensibility analysis, we perturbed the economic values of the blocks. For this, we considered one independent random variable $\alpha \sim Triangular(-1, 0.0, 1)$ per block. Thus, if the economic value of the block was v_b, its scenario in scenario k became $v_{bk} = v_b \pm 20\% \cdot \alpha \cdot v_b$. We generated scenarios $k = 1, 2, \dots, K = 100$ with this method.

Table 6 presents the results obtained for the constrained drift location case. Similar results are obtained using the free drift location variant. For each case (A1 and B1), the table presents how the economic value and tonnage of optimal solutions varied. Four values are used for comparison: the base case (obtained using the

original economic values), minimum, maximum, and average over all the scenarios. As can be seen from the results, the difference between the base case and the average of the scenarios is small (less than 0.4% in the worst case). Moreover, the total variability is very small. Indeed, the coefficient of variability is less than 0.3%. This suggests that the solution obtained for the base case is robust regarding the perturbation considered.

Table 6 - Summary of results of sensibility analysis

		Scenarios				Statistics		
		Base Case $[1]$	Min	Max	Avg $[2]$	Base vs Avg. $\frac{\lvert[1]-[2]\rvert}{[1]}$	Stan. Dev. $[4]$	Coef. Var. $\frac{[4]}{[2]}$
A1	Value (KUSD)	67,338	67,006	67,874	67,350	0.02%	163	0.24%
	Tonnage (Kt)	2,835	2,835	2,835	2,835	0.00%	-	0
B1	Value (KUSD)	413,089	410,094	413,992	412,883	0.05%	283	0.07%
	Tonnage (Kt)	38,739	37,850	39,186	38,594	0.38%	1	0.00%

To analyze the spatial variability of the solutions, we performed a similarity analysis using the Jaccard similarity index (Jaccard, 1912). Given two discrete sets A and B, the index is calculated as $JACC(A,B) = \frac{|A \cap B|}{|A \cup B|}$. The maximum value of the index occurs when $A = B$, in which case $JACC(A,B) = 1$. Conversely, if $A \cap B = \emptyset$, then $JACC(A,B) = 0$, which is the minimum possible value of the index.

We applied the Jaccard index considering the optimal layouts as discrete sets of blocks. In each scenario, a block is marked with a 1 if it is part of the selected layout, and 0 otherwise. Computations are done only over the sets of blocks that were selected by at least one layout.

Figure 10 and Figure 11 present the results for A1 and B1, respectively. The left sides of the figures display the histograms over the 100 scenarios and the right sides present the dispersion of these results. In both cases, we observe that the set of blocks selected for extraction is stable because in most scenarios (95% of scenarios of A1 and 100% of scenarios of B1) the Jaccard index indicates that a fraction of 0.95 of the blocks of A1 selected for extraction remained selected. In fact, in the few cases of A1 (5/100) where the index was below 0.95, it remained above 0.65.

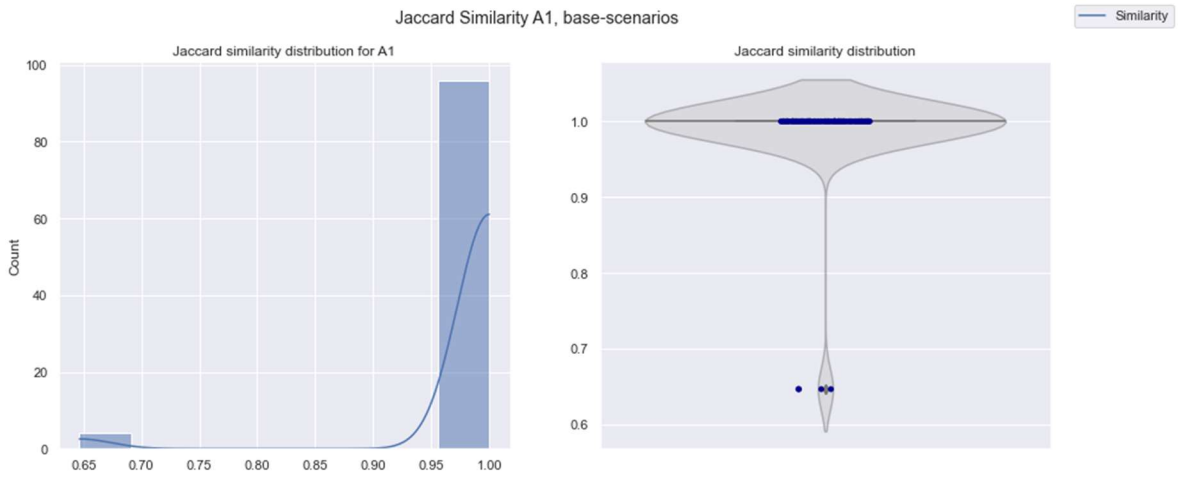


Figure 10: Jaccard index distribution and dispersion. Left = Case A1. Right = Case B1.

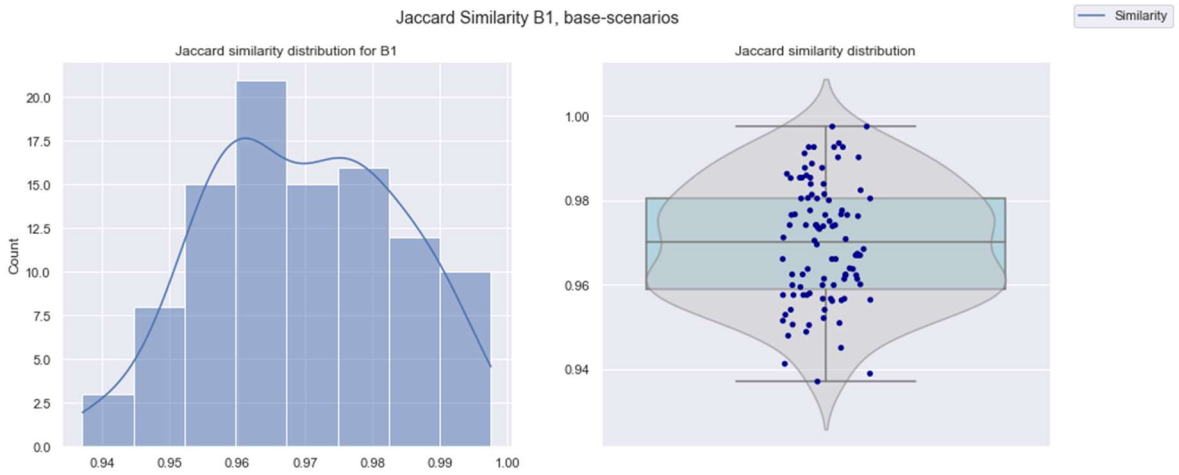


Figure 11: Jaccard index distribution and dispersion for B1.

4 Conclusions

This paper addresses the problem of determining the best underground layout in a sublevel stoping mine considering the location of production drifts and stopes such that the overall economic value is maximized. For this, similarly to previous studies, we follow a two-step methodology in that we (i) generate valid stopes, and (ii) search for the optimal set of selected stopes (maximum total economic value). However, we make several contributions in both stages.

In the first stage, where a set of valid stopes is generated and evaluated, we consider the drawbells and production drifts as part of the geometry of the stope, thus we can more accurately estimate the costs and income. Moreover, we observe that during the process of generating the stopes, it is possible to evaluate other attributes, which permits stope stability to be analyzed. With this information, we can remove stopes that are geomechanically unstable in the first stage, so they are not included in the second stage. This ensures that the stopes selected in the optimal layout are stable and reduces the size of the optimization process of the second stage.

In the second stage of the methodology, the approach followed by others has been to run a heuristic method (that does not ensure optimality) or to solve an optimization problem (which several authors acknowledge is difficult to solve). We follow the second approach and propose a binary optimization model. The model has several advantages over other approaches:

- It is very flexible. In the paper we study two variations implementing different degrees of freedom for the location of the drifts: The free drift location, where production drifts can be located in any position (provided that they are compatible), and the constrained drift location, where drifts must be grouped into sublevels that are selected by the model.
- It can be solved quickly. Contrary to other approaches that report difficulty finding optimal solutions in reasonable time, the computational times required in our approach are practical even when using free academic solvers.
- It has good theoretical properties. In the case of the constrained drift location, we show that the problem can be reduced to the computation of a shortest-path problem in an auxiliary graph constructed ad-hoc. For the free drift location, we propose a fast heuristic that can find relatively good solutions in short time frames.

We apply the methodology to two mines, for each of which two block models were available (the difference being the size of the blocks), giving a total of four instances ranging from about 84,000 blocks to more than 3.2 million blocks.

In the first stage of the methodology, the set of valid stopes is generated. A first relevant result here is that eliminating stopes with negative value or instability reduces the number of potential stopes between 90% and 97%. This is significant because it speeds up the optimization process in the second stage.

In the second stage of the methodology, we run our mathematical model to show that it generates practical geometries in a relatively short time. Indeed, computational times using free academic solvers to find optimal solutions are below 45 minutes in the larger instances. When comparing the two variations of the model, as expected, the layout proposed by the free drift location variation of the model generates higher values, but the geometry is more complex if compared to the layout obtained when applying the constrained drift location. This suggests, for example, that seeking some combination of the variations, using first the free drift location to have a better understanding of the best location of stopes in 3-D space and then potentially dividing the domain into zones in which the constrained variation could be applied . Finally, in the case of the constrained drift locations, it is possible to apply a shortest-path algorithm which finds the optimal solution in less than one second.

To analyze the robustness of the results, we performed a sensibility analysis of the optimal layouts, We considered perturbations of the economic values of the stopes and analyzed the variability of the value, tonnage and geometry of the stopes obtained by our model. The results show that the variation in value and tonnage of the optimal solutions is below 1%. Moreover, the blocks selected for extraction remain mostly the same, as in most of the scenarios the number of blocks that changed their status was below 5%.

Future work will include applying the model to more mines and evaluating variations of the model. The sensibility analysis also suggests that it would be interesting to develop a stochastic version of the model in terms of market or geological uncertainty and apply strategies to solve the corresponding model.

References

Alford, C. (1995). Optimisation in underground mine design. *Proceedings of the 25th Application of Computers and Operation Research in the Mineral Industry*, 213–218.

Ataee-Pour, M. (2005). A critical survey of the existing stope layout optimization techniques. *Fiziko-Tekhnicheskie Problemy Razrabotki Poleznykh Iskopaemykh*, *41*(5), 62–82.

AtlasCopco. (2007). *Mining methods in underground mining* (Atlas Copco).

Bai, X. (2013). *Optimization of underground stope with network flow method*. University of Montreal.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, *16*, 87–90.

Cawrse, I. (2001). Multiple pass floating stope process. *Proceedings of Stratgic Mine Planning Conference*, 87–94.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Erdogan, G., & Yavuz, M. (2017). Application of Three Existing Stope Boundary Optimisation Methods in an Operating Underground Mine. *IOP Conference Series: Earth and Environmental Science*, *95*(4). https://doi.org/10.1088/1755-1315/95/4/042077

Ford, L. (1956). *Network Flow Theory*. RAND Corporation. https://www.rand.org/pubs/papers/P923.html

Garey, M. R., & Johnson, D. S. (1978). ` Strong ' ' NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the ACM*, *25*(3). https://doi.org/10.1145/322077.322090

Grieco, N., & Dimitrakopoulos, R. (2007). Managing grade risk in stope design optimisation: probabilistic mathematical programming model and application in sublevel stoping. *Mining Technology*, *116*(2). https://doi.org/10.1179/174328607X191038

Hou, J., Xu, C., Dowd, P. A., & Li, G. (2019). Integrated optimisation of stope boundary and access layout for underground mining operations. *Mining Technology: Transactions of the Institute of Mining and Metallurgy*, *128*(4), 193–205. https://doi.org/10.1080/25726668.2019.1603920

Jaccard, P. (1912). The distribution of the flora in the alpine zone 1. *New Phytologist*, *11*(2), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Jalali, E. E., & Ataee-pour, M. (2004). A 2-D dynamic programming algorithm to optimize stope boundaries. *Proceedings of Mine Planning and Equipment Selection*, 45–52.

Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, *28*(3), 497–520. https://doi.org/https://doi.org/10.2307/1910129

Little, J., Knights, P., & Topal, E. (2013). Integrated optimization of underground mine design and scheduling. *Journal of the Southern African Institute of Mining and Metallurgy*, *113*(10), 775–785.

Mathews, K. E., Hoek, E., Wyllie, D. C., & Stewart, S. (1981). *Prediction of Stable Excavation Spans for Mining Depths below 1000m in Hard Rock*.

Mawdesley, C., Trueman, R., & Whiten, W. J. (2001). Extending the Mathews stability graph for open–stope design. *Mining Technology*, *110*(1). https://doi.org/10.1179/mnt.2001.110.1.27

Nelis, G., Gamache, M., Marcotte, D., & Bai, X. (2016). Stope optimization with vertical convexity constraints. *Optimization and Engineering*, *17*(4), 813–832. https://doi.org/10.1007/s11081-016-9321-6

Nikbin, V., Ataee-pour, M., Shahriar, K., & Pourrahimian, Y. (2020). A 3D approximate hybrid algorithm for stope boundary optimization. *Computers & Operations Research*, *115*, 104475. https://doi.org/10.1016/j.cor.2018.05.012

Nikbin, V., Ataee-pour, M., Shahriar, K., Pourrahimian, Y., & MirHassani, S. A. (2019). Stope boundary optimization: A mathematical model and efficient heuristics. *Resources Policy*, *62*(June), 515–526. https://doi.org/10.1016/j.resourpol.2018.10.007

O'Sullivan, D., & Newman, A. (2015). Optimization-based heuristics for underground mine scheduling. *European Journal of Operational Research*, *241*(1). https://doi.org/10.1016/j.ejor.2014.08.020

Ovanic, J., & Young, D. (1999). Economic optimization of open stope geometry. *Proceedings of the 28th Application of Computers and Operation Research in the Mineral Industry*, 855–862.

Potvin, Y. (1988). Empirical Open Stope Design in Canada. In *Retrospective Theses and Disertations 1919-2007*.

Sandanayake, D. S. S., Topal, E., & Asad, M. W. A. (2015). Designing an optimal stope layout for underground mining based on a heuristic algorithm. *International Journal of Mining Science and Technology*, *25*(5), 767–772. https://doi.org/10.1016/j.ijmst.2015.07.011

Sourineni, F. W. (2010). The stability graph after three decades in use: Experiences and the way forward. *International Journal of Mining, Reclamation and Environment*, *24*(4), 307–339.

Topal, E., & Sens, J. (2010). A new algorithm for stope boundary optimization. *Journal of Coal Science and Engineering*, *16*(2), 113–119.

Appendix A – Demonstration of theoretical results

A.1 Reduction of $(1 - SDP)$ to the shortest-path problem

For this, let us define $y^-(\ell), y^+(\ell)$ as the y coordinate of the first (minimum) and last (maximum) slice that belong to the stope (recall that the drifts follow the y direction). We see that stope $\ell_2 \in \Lambda$ can be constructed after a stope $\ell_1 \in \Lambda$ if $y^+(\ell_1) \leq y^-(\ell_1)$, and $\{\ell_1, \ell_2\} \notin \Gamma$.

Now, consider the set of nodes $\Lambda' = \{s, t\} \cup \Lambda$ where s, t are two auxiliary nodes, and the set of arcs $\Gamma' = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, where:

- $\Gamma_1 = \{(s, \ell) : \ell \in \Lambda\}$, i.e., vertex s is connected to all stopes,
- $\Gamma_2 = \{(\ell_1, \ell_2) \in \Lambda \times \Lambda : \ y^+(\ell_1) \leq y^-(\ell_2) \wedge \{\ell_1, \ell_2\} \notin \Gamma\}$, which connects compatible stopes, and
- $\Gamma_3 = \{(\ell, t) : \ell \in \Lambda\}$, that is, all stopes are connected to t.

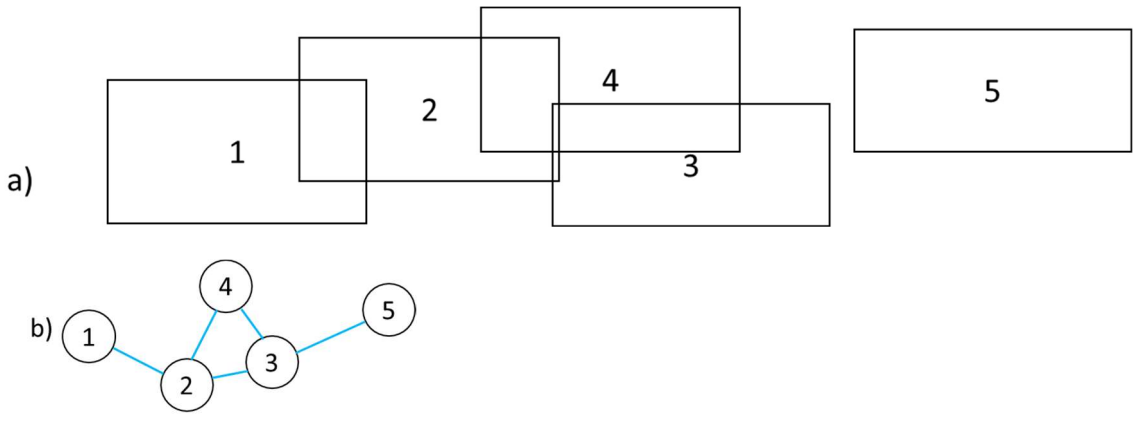
This defines an acyclic directed graph $G' = (\Lambda', \Gamma')$. We observe that arcs in G' either end in a slope ℓ or in the auxiliary vertex t; therefore, we can define the weight function on the arcs as

$$v'(a, b) = \begin{cases} v(b) & \text{if } b = \ell \text{ for some } \ell \in \Lambda \\ 1 & \text{if } b = t \end{cases}$$

Finally, we observe that any independent set $X \subset \Lambda$ in G with weight $V = \sum_{\ell \in X} v(\ell)$ induces a path in G' which starts in s, goes through all stopes in X and ends in t. This path has $V + 1$ arcs and has a length $L = 1 + \sum_{\ell \in X} v(\ell) + 1$. It follows that finding the longest path in G' is equivalent to finding an independent set in G. However, as G' is acyclic, this can be solved using shortest-path algorithms, which can be done in polynomial time (Sedgewick & Wayne, 2011).

The construction is illustrated in Figure 12(a), (b) and (c). In (a) a small conceptual case is presented consisting of 5 stopes, where stopes 3 and 4 are incompatible because they overlap. Stopes 3 and 5 are also incompatible because they are too close, and stope 2 is incompatible (due to overlapping) with stopes 1, 3 and 4. These incompatibilities are summarized in (b) which presents the set of stopes $\Lambda = \{1,2,3,4,5\}$ and the set of incompatibility $\Gamma = \{\{1,2\}, \{2,3\}, \{2,4\}, \{3,4\}, \{3,5\}\}$. Finally, (c) shows the auxiliary construction: the vertices $\Lambda' = \{s, 1,2,3,4,5, t\}$ and arcs $\Gamma' = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ where $\Gamma_1 = \{(s, \ell) : \ell \in \Lambda\}$ is in red, $\Gamma_2 = \{(1,3), (1,4), (1,5), (2,5), (4,5)\}$ is in purple and $\Gamma_3 = \{(\ell, t) : \ell \in \Lambda\}$ in shown in green.

In the example, we see that if each stope has a value $v(\ell) = 1$ then the set of stopes to be constructed (according to (b)) is the set $I = \{1,2,4\}$, which is an independent set in $G = (\Lambda, \Gamma)$. Set I corresponds, in fact, to the longest path in the directed $G' = (\Lambda', \Gamma')$ starting at s, which is $\pi = \{s, 1,2,4, t\}$.



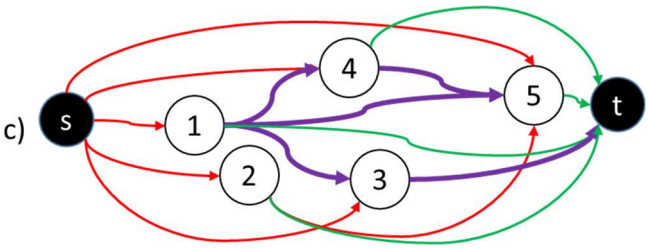


Figure 12 - (a) A conceptual set of stopes in a drift. (b) Incompatibility graph for stope construction (edges in blue represent the elements of Γ). (c) Associated compatibility graph for stope construction.

Notice that, in fact, adding vertex t is not necessary for the construction; however, it is convenient to do so because it simplifies the extension to the general case. More importantly, we observe that the reduction of $(1-SDP)$ to a shortest path works because of the following *transitivity condition*: if ℓ_1 is compatible with ℓ_2 and ℓ_2 is compatible with ℓ_3 then ℓ_1 is compatible with ℓ_3.

A.2 Reduction of $(n-SDP)$ to the shortest-path problem (constrained drift case)

In this section, we address the general case by extending the construction of the one-dimensional case, i.e., we enumerate the drifts and establish a compatibility graph such that an independent set of drifts corresponds to a path in the graph.

Let $m = 1, 2, \dots, |XZ|$ be the enumeration of the drifts in growing order z coordinates and then in ascending order of x coordinate so if m corresponds to (x, y) and m' to (x', y') are such that $m < m'$, then either $z = z', x < x'$ or $z < z'$. Given a drift with index m, we denote $x(m), z(m)$ its coordinates.

As each m defines an instance $(1-SDP)$ with a set of stopes Λ_m and incompatibility edges Γ_m, we denote as $G_m = (\Lambda'_m, \Gamma'_m)$ the auxiliary directed graph constructed as in the previous section, and v'_m the corresponding weight function, defined over the arcs of G_m. We then use the graphs G_m to construct a directed graph $G'' = (V'', \Gamma'')$, as follows. The set of vertices is $V'' = \{s\} \cup \bigcup_m \Lambda'_m$. The set of arcs is $\Gamma'' = \{(s, s_m)\}_m \cup \bigcup_m \Gamma''_m \cup \{(t_{m_1}, s_{m_2}) : m_1 < m_2 \wedge \{m_1, m_2\} \notin \hat{\Gamma}\}$. The weight function is $v''(a,b) = \begin{cases} 1 & a = s \text{ or } b = t \\ v'_m(a,b) & a \neq s, t \end{cases}$.

A small example of the construction is depicted in Figure 13, which shows a conceptual case consisting of four drifts, A, B, C and D, with B being incompatible with A and B. On the right, the corresponding graph structure shows how the graphs for individual drifts are connected. The internal structure of the drifts is omitted to simplify the figure, but it corresponds to the construction for the one-dimensional case.

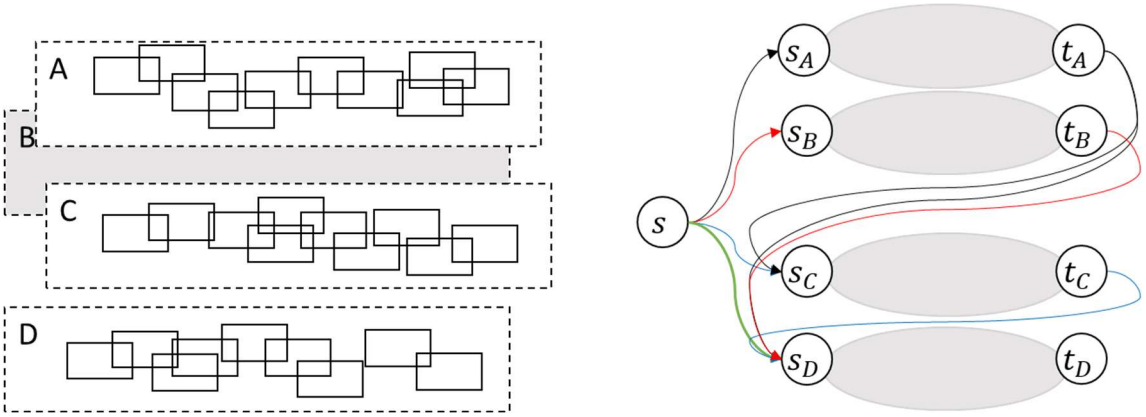


Figure 13 – Left: An example consisting of four drifts (A, B, C, D) where B is incompatible with A and C. Right: graph for computing the set of stopes with maximum value. (Internal graph structure for each drift is omitted for simplicity.)

We observe that in the constrained drift location case, given drifts $m_1 < m_2 < m_3$ such that $(m_1, m_2), (m_2, m_3) \in \Gamma''$ (that is m_1 is compatible with m_2 and m_2 is compatible with m_3). Then, either

- $z(m_1) = z(m_3)$ are at the same level, thus, drift m_2 is in between m_1 and m_2, but all stopes in m_1 are compatible with all stopes in m_2 (they are far enough), and all stopes in m_2 are compatible with all stopes in m_3 (they are far enough). Then all stopes in m_1 are far enough from all stopes in m_3, or
- $z(m_1) > z(m_3)$, in which case $z(m_1) - z(m_3) > D_z$.

In both cases we have that m_1 is compatible with m_3, i.e., $(m_1, m_3) \in \Gamma''$. That is, G'' satisfies the transitivity condition, and therefore a path defines an independent set of G'' and the length of the path corresponds to the economic value of the selected stopes. Moreover, the graph has no cycles; therefore, computing the longest path can in turn be reduced to a shortest-path problem.